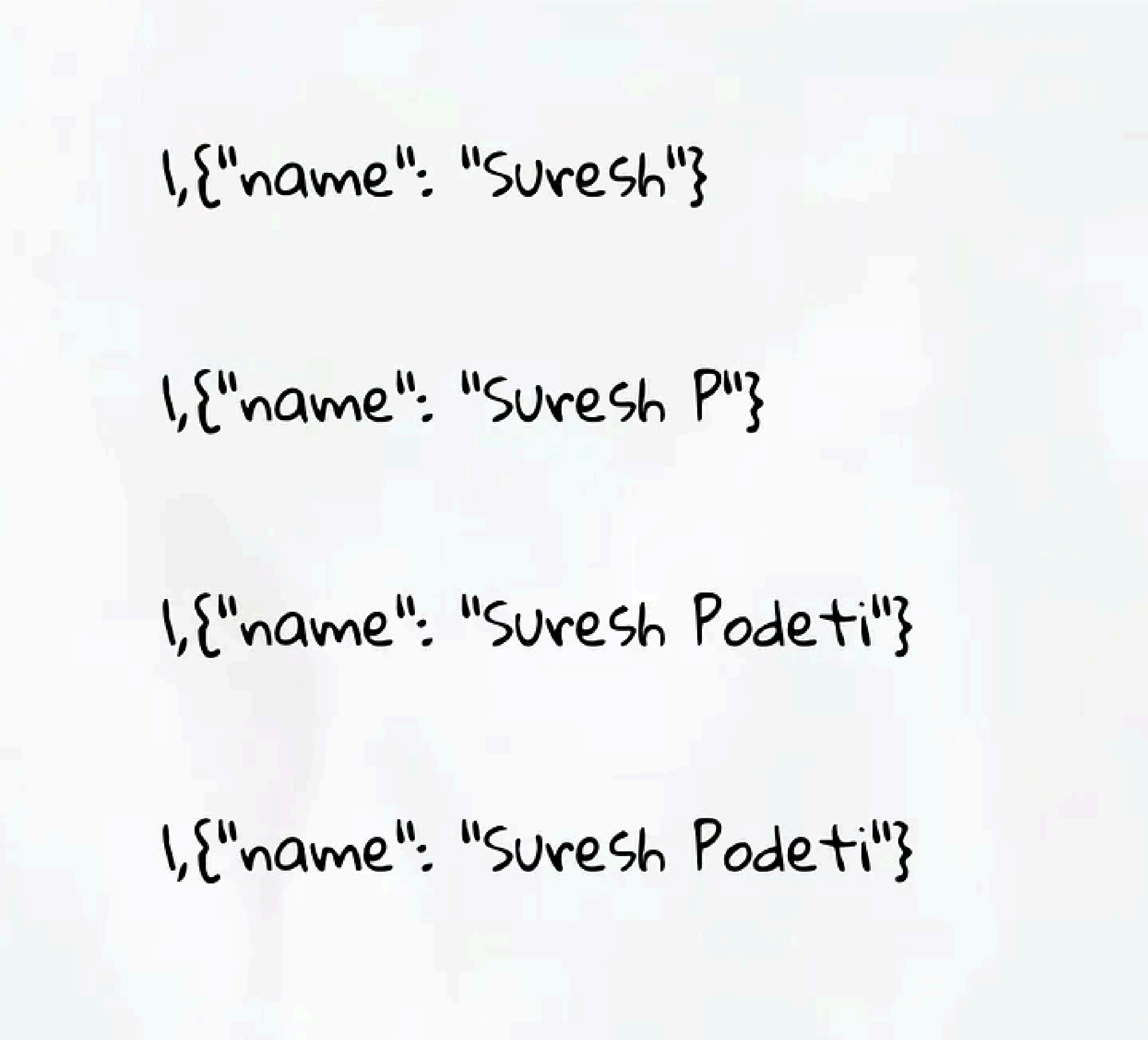
Log compaction and Merging

Suresh Podeti

2 min read · May 15, 2023

Photo by Volodymyr Hryshchenko on Unsplash

As we have seen in the previous articles database stores data by writing to an **append-only log file** as shown in Fig 1. below:



```
1,{"name": "Suresh"}

1,{"name": "Suresh P"}

1,{"name": "Suresh Podeti"}

1,{"name": "Suresh Podeti"}
```

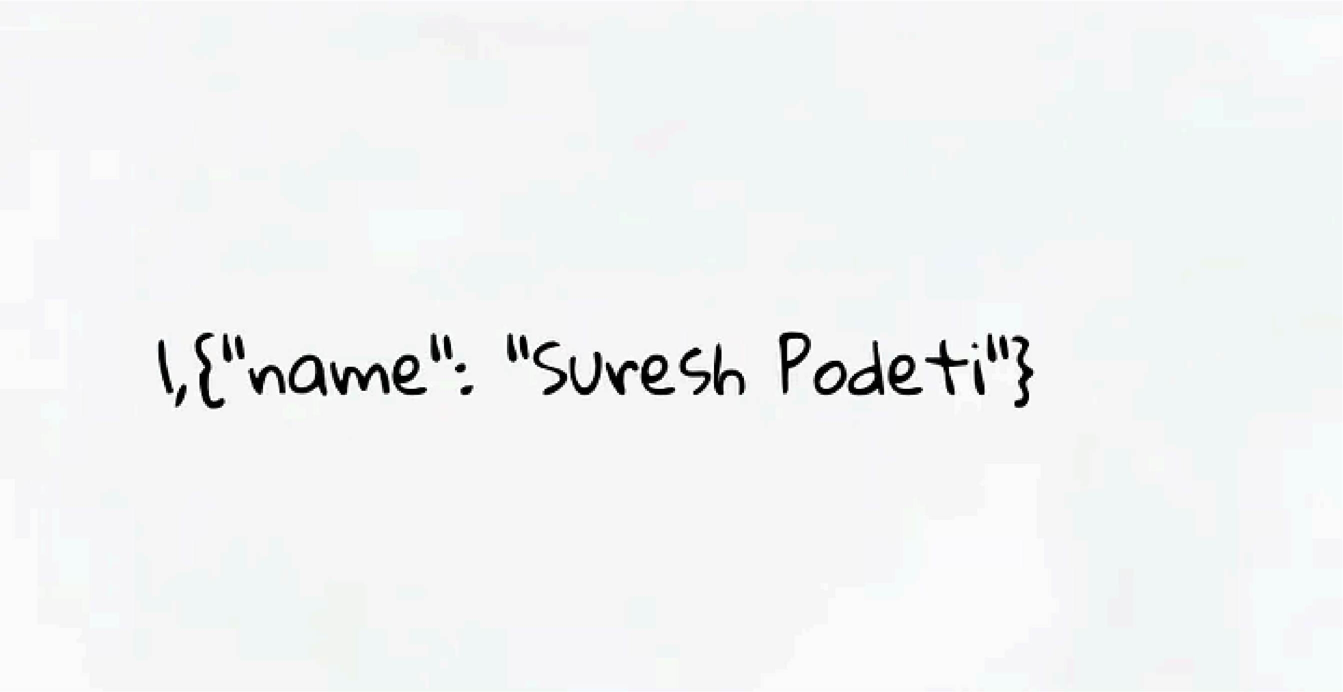
Fig 1. An append-only log file containing the key-value pairs

Every operation to the database appends an entry in the log file, and stored as sequence of key-value pairs. These pairs **appear in the order that they were written**, and **values later in the log take precedence over values for the same key earlier** in the log.

If we keep on appending to the log file **eventually** we get to a point where we **run out of disk space**. A good solution is to **break the log into segments of a certain size** by closing a segment file when it reaches a certain size, and making subsequent **writes to a new segment file**. We can then perform **compaction** on these segments.

Compaction

Compaction means **throwing away duplicate keys** in the log, and **keeping** only the **most recent update** for each key. Applying compaction to the above mentioned example we will have only one entry for the key 1 as shown below:



```
1,{"name": "Suresh Podeti"}
```

Fig 2. Compacted append-only log file

Merging

Moreover, since compaction often makes segments much smaller, we can also **merge** several segments together at the same time as performing the compaction. Segments are never modified after they have been written, so merged segment is written to a new file.

The compaction and merging of frozen segments can be done in a **background thread**, and while it is going on, we can still continue to serve read and write requests as normal, using the old segment files. After the merging process is complete, we switch read requests to using the new merged segment instead of the old segments-and then the old segment files can simply be deleted.

Entire compaction-merging process can be visualised as shown in the Fig 3. below:

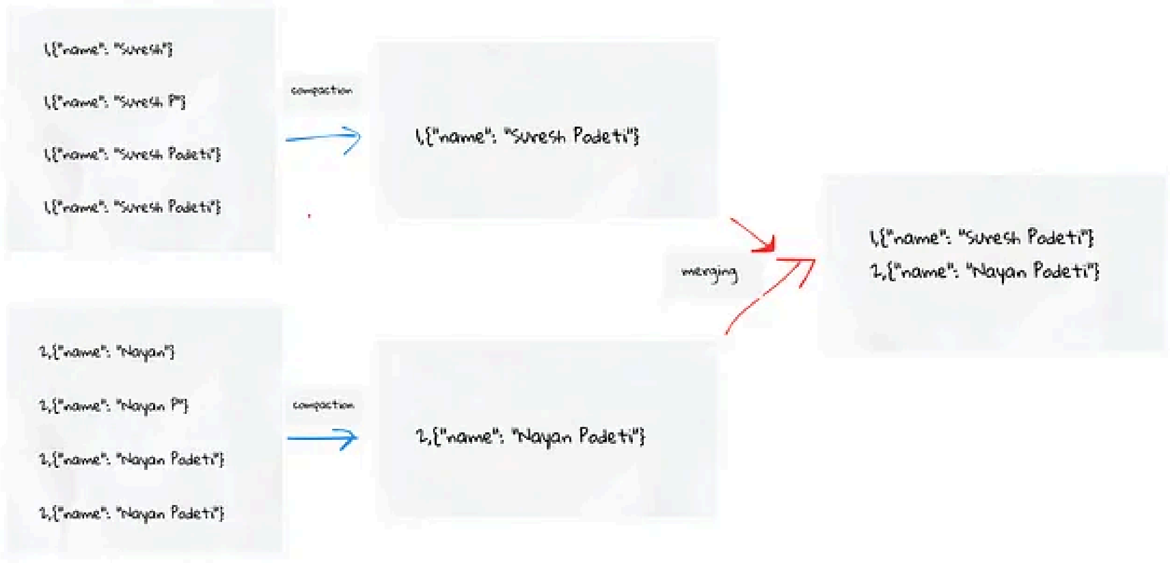

1,{"name": "Suresh"}
1,{"name": "Suresh P"}
1,{"name": "Suresh Podeti"}
1,{"name": "Suresh Podeti"}
compaction
1,{"name": "Suresh Podeti"}
merging
1,{"name": "Suresh Podeti"}
2,{"name": "Nayan Podeti"}
2,{"name": "Nayan"}
2,{"name": "Nayan P"}
2,{"name": "Nayan Podeti"}
2,{"name": "Nayan Podeti"}
compaction
2,{"name": "Nayan Podeti"}

Fig 3. Append-only log compaction-merging in nutshell

Reference:

1. O'Reilly designing data-intensive applications by Martin Kleppmann, Chapter 3: Storage and Retrieval

Compaction Merging Database Storage Engine Database Index

Edit profile

Written by Suresh Podeti

1.2K Followers

Aspiring Software Architect | SDE III (Golang) @JungleeGames| Ex-Byju's | Co-founder @Rupant Tech. | Ex-Synopsys | I.I.T Kharagpur | 7+ Years of Experience